

APPENDIX  
(SOURCE CODE)

/\*-----

Text Rain Source Code

written by Camille Utterback

-----\*/

#include <windows.h>

#include <stdlib.h>

#include <string.h>

#include <vfw.h>

#include <time.h>

#include <wingdi.h>

#define MAXNUMLINES 25

//the max number of lines that can be entered at one time (this is just a practical limmit for testing  
- no real reason to keep it this low

#define MAXNUMLETTERS 60

0571011-012501

[illegible]

```
int y;
```

int rate;

COLORREF fadeColor;

}

DROP;

//--LINE struct - an array of drops, there will be one for each line

typedef struct

{

DROP dropline[MAXNUMLETTERS]; //should call this dropArray

//int dropArrayLength OR LINEINFO LineInfo

}

LINE;

//--LINEINFO struct - info about each line (should really be part of LINE)

typedef struct

{

bool move;

bool fade;

COLORREF color;

//int length;

TOGETHER = F.T.O.T.260

```
/* -----Declare all functions -----*/
```

```
void Run(HWND hwnd);           //prints everything from the buffer to the screen
```

```
int FindDriver(char driverstring[80], int stringlength);
```

```
int SetArray(char[], COLORREF, DROP[]); //returns length of array
```

```
LRESULT CALLBACK VideoCallbackProc(HWND, LPVIDEOHDR); //video callback
function
```



/\*-----Declare all global variables -----\*/

//-----STRING

LINE Lines[MAXNUMLINES]; //an array of LINE structs (which are each an array of DROP structs

LINEINFO LineInfos[MAXNUMLINES];

int LineLengths[MAXNUMLINES]; //array that will hold the letter length of each line in the poem

//could incorporate this into lineinfos.

COLORREF colors[]={GREEN,PINK,GOLD,BLUE,RED};

int NumColors;

int gNumLines;

//-----GENERAL

09771011:012601

int cxClient, cyClient; // window dimensions

TEXTMETRIC tm;

int cxChar, cyChar, cxCaps; //x and y dimensions of text

char buffer[MAXREAD];

int CursorValue=0;

int ActiveLineIndex=0;

int HoleWidth=4;

int HoleHeight=4;

int HoleL=309; //int HoleL=(640)-(HoleWidth/2);

int HoleT=237; //int HoleT=(480)-(HoleHeight/2);

int HoleGrey=0;

int DarkThresh=180;

bool firstpaint=1;

//-----WINDOW/OBJECTS .

007401-012601  
10520:TT02200

HWND hwnd; //main window

HWND ghWndCap; //capture window

HDC hdc;

HDC viddc; //memdc for video image

HBITMAP tempbitmap2;

HDC backwardsdc; //memdc for flipping video image

HBITMAP BackwardsBitmap;

HDRAWIDIB hdd; //to decompress captured info

//-----VIDEO

HGLOBAL hgout;

DWORD dwsiz;

LPVOID pformatbmiv;

LPBITMAPINFO pformatbmi;

unsigned char\* pbits;

09771011-012601  
FDSTO-TOTZ60



/\*-----\*/

int WINAPI WinMain(HINSTANCE hInstance, HINSTANCE hPrevInst, PSTR szCmdLine, int  
iCmdShow)

{

static char szAppName[] = "Text Rain";

MSG msg;

WNDCLASSEX wndclass;

int i;

/\*define a window class\*/

wndclass.cbSize = sizeof(wndclass);

wndclass.style = 0; /\*0 was the default style\*/

wndclass.hInstance = hInstance; /\*handle to this instance\*/

wndclass.lpszClassName = szAppName; /\*window class name\*/

wndclass.lpfnWndProc = WindowProc; /\*window function\*/

wndclass.hIcon = LoadIcon(NULL, IDI\_APPLICATION); /\*icon style\*/

09771011:012601

```
wndclass.hIconSm = LoadIcon(NULL, IDI_APPLICATION);

wndclass.hCursor = LoadCursor(NULL, IDC_ARROW); /*cursor style*/

wndclass.lpszMenuName = NULL; /*no menu*/


wndclass.cbClsExtra = 0; /*no extra*/

wndclass.cbWndExtra = 0; /*info needed*/


wndclass.hbrBackground =(HBRUSH) GetStockObject(WHITE_BRUSH);


RegisterClassEx (&wndclass);


hwnd = CreateWindow(
    szAppName, /*name of window class*/
    NULL, /*title */
    WS_POPUP, /*window style*/
    0, /*x coordinate*/
    0, /*y coordinate*/
    640, /*width coordinate*/
    480, /*height coordinate*/
    NULL, /* or HWND_DESKTOP no parent window*/
    NULL, /*menu handle (if parent) - child id if window is a child*/
```

hInstance, /\*handle of this instance of the program\*/

NULL /\*no additional arguments\*/

);

/\*windows sends WM\_CREATE msg to wndProc while processing createwindow \*/

/\*display window\*/

ShowWindow(hwnd, iCmdShow); /\*windows sends WM\_SIZE,  
WM\_SHOWWINDOW\*/

UpdateWindow(hwnd); /\* sends WM\_PAINT message to windProc \*/

//-----

while (TRUE)

{

if (PeekMessage (&msg, NULL, 0,0, PM\_REMOVE))

{

if (msg.message == WM\_QUIT)

break;

TranslateMessage(&msg);

DispatchMessage(&msg); /\* sends msg to windows which calls wndProc

with it\*/

00771011-012601

```

char szTextFile[]="words.txt";      //name of file to read from

static char sReadBuffer[MAXREAD];    //our buffer to read stuff into

static char szTempBuffer[MAXREAD];

DWORD charsreadok;


int i,j,linenum,t,g;

int MaxLineLength;

int ColorIndex;


switch(iMsg)
{

case WM_CREATE:

    //-----get device context

    hdc=GetDC(hwnd);


    //-----create VIDDC

    viddc=CreateCompatibleDC(hdc);

    tempbitmap2=CreateCompatibleBitmap(hdc, 640, 480);

    SelectObject(viddc,tempbitmap2); //do I even need to do this?

```

SetBkMode(viddc, TRANSPARENT);

//-----CREATE BACKWARDSDC

backwardsdc=CreateCompatibleDC(hdc);

BackwardsBitmap=CreateCompatibleBitmap(hdc, 640, 480);

SelectObject(backwardsdc,BackwardsBitmap); //enlarge dc from 1 monochrome

pixel

//-----CREATE HDRAWDIBs

hdd=DrawDibOpen();

//-----get Text Metrics

GetTextMetrics(hdc,&tm);

cxChar=tm.tmAveCharWidth+1;

cyChar=tm.tmHeight+tm.tmExternalLeading+1;

cxCaps= (tm.tmAveCharWidth \* 3)/2+1; //cxChar=avecharwidth

//-----READ IN USER'S asci STRING and SET ARRAY

0971011-01601  
10921011-01601

if(!((ReadInTextFromFile(szTextFile,sReadBuffer,MAXREAD,&charsreadok))  
&& (strlen(sReadBuffer)>1))){

//---readtext didn't work or string is only one char long

MessageBox( hwnd, // handle of owner window  
"can't read in text", // address of text  
in message box  
NULL, // address of title of message  
box  
MB\_OK // style of message box  
);

CleanUp();  
PostQuitMessage(0); //user clicked close, puts WM\_QUIT  
message in the message queue

}else{

//---readtext in from file OK, parse through it.

MaxLineLength=MAXNUMLETTERS; //length of longest line in textrain  
arrays.

NumColors=(sizeof(colors))/(sizeof(COLORREF));//5

ColorIndex=0;

linenum=0;

i=0;

j=0;

[illegible]

## happens i

```
if(sReadBuffer[i]=='\t'){
```

```
for(t=0;t<5;t++){
```

```
szTempBuffer[j]=' ';
```

```
j++; //don't increment i here - it
```

}

}

```
}else{
```

```
//---all other chars
```

```
if(sReadBuffer[i]!='\n'){
```

```
//--not a newline, add to array
```

```
szTempBuffer[j]=sReadBuffer[i];
```

0971011-012601  
109210-11012601

```
j++;
```

```
}
```

```
}
```

```
i++; //tab, newline, or normal char
```

```
} else { //end if(j<MaxLineLength)&&(sReadBuffer[i]!='r')
```

```
    //j> maxlenlength or we're at a cr
```

```
    //in either case - we're at the end of the line
```

```
    if(j>=MaxLineLength){ //move I to next cr
```

```
        while((sReadBuffer[i] != '\r') && (i<(charsreadok)))
```

```
{
```

```
            i++;
```

```
}
```

```
}
```

```
i++; //i= cr whether or not we hit maxline, move to next
```

letter

```
//---SET ARRAY
```



times)

//send temparray to SetArray (or do this numrepeatlines

//--- set up LineInfos struct for this line

if(linenum<2)

LineInfos[linenum].move=1;

else

LineInfos[linenum].move=0;

LineInfos[linenum].fade=0;

LineInfos[linenum].color=colors[ColorIndex];//RGB(200,100,155)

if(ColorIndex<(NumColors-1))

ColorIndex++;

else

ColorIndex=0;

//--- send info to SetArray

LineLengths[linenum] = SetArray(szTempBuffer,  
LineInfos[linenum].color, Lines[linenum].dropline);

097101:04504  
F0920:F0F260

struct for each letter,

//this takes a string of text Text[i], fills a DROP

(basically the strlen)

//and returns the num of DROP structs filled

...

for(t=0;t<MaxLineLength;t++)

szTempBuffer[t]=' ';

j=0;

linenum++;

}

}

gNumLines=linenum; //this will be the actual NUMBER of lines, not the  
last index num.

}//end else for reading in file

0071010304

//-----CREATE CAPTURE WINDOW

ghWndCap = capCreateCaptureWindow((LPSTR)"Capture Window",  
WS\_CHILD|WS\_VISIBLE, 0,0,600,440,(HWND) hwnd, (int)0);

//-----register callback functions

capSetCallbackOnVideoStream(ghWndCap,&VideoCallbackProc);

return 0;

case WM\_SIZE:

**/\*iParam that gets passed with this message contains  
the new width of the client window in the lowWord  
and the new height of the client window in the HiWord\*/**

cxClient=LOWORD(iParam);

cyClient=HIWORD(iParam);

09771011-012601

//-----connect to the driver

```
//if(capDriverConnect(ghWndCap, FindDriver(QuickCamString,
(int)strlen(QuickCamString)))){
```

```
if(capDriverConnect(ghWndCap, FindDriver(ATIString,
(int)strlen(ATIString)))){
```

//-----get and set streaming vid parameters

```
capCaptureGetSetup(ghWndCap,&CapParms, sizeof(CapParms));
```

```
CapParms.fCaptureAudio=0; //no audio
```

```
CapParms.dwRequestMicroSecPerFrame=22222;
```

```
CapParms.fAbortRightMouse=0; //no action for rt mouseclick
```

```
CapParms.fAbortLeftMouse=0; //no action for lft mouseclick
```

```
capCaptureSetSetup(ghWndCap,&CapParms, sizeof(CapParms));
```

```
capPreviewScale(ghWndCap,1); //stretch preview to size of capture window
```

```
return 0;
```

```
}
```

109271011-012601

else{

MessageBox( hwnd, "can't connect to capture driver", NULL,  
MB\_OK|MB\_SYSTEMMODAL );

CleanUp();

//-----put WM\_QUIT message in the message queue (your program won't  
get this back)

PostQuitMessage(0);

return 0;

}

case WM\_PAINT:

if (firstpaint==1){

BitBlt(viddc, 0, 0, cxClient, cyClient, hdc, 0, 0, SRCCOPY); //save a copy

firstpaint=0;

}

return 0;

case WM\_LBUTTONDOWN:

09771011-012601  
T092T0-T0T4250

//---video SOURCE dialog - BRIGHTNESS,CONTRAST etc for qc

//if cursor is invisible

if(CursorValue<0) //will be -1 if invis

CursorValue=ShowCursor(1);//show cursor

capDriverGetCaps(ghWndCap, &CapDrvCaps, sizeof(CAPDRIVERCAPS));

if (CapDrvCaps.fHasDlgVideoSource)

capDlgVideoSource(ghWndCap);

return 0;

case WM\_RBUTTONDOWN:

//---video FORMAT dialog box - IMAGE SIZE and QUALITY for qc

//if cursor is invisible

if(CursorValue<0) //will be -1 if invis

09771011-01504  
105210-1102250



AperatureTest();

TextOut(hdc, 30, 30, buffer, wsprintf(buffer, "loc(L,T) = %d, %d  
w=%d h=%d", HoleL, HoleT, HoleWidth, HoleHeight));

TextOut(hdc, 30, 70, buffer, wsprintf(buffer, "grey = %d",  
HoleGrey));

TextOut(hdc, 30, 70, buffer, wsprintf(buffer, "gNumLines = %d",  
gNumLines));

break;

case VK\_DOWN:

BitBlt(hdc, 0, 0, cxClient, cyClient, viddc, 0, 0, SRCCOPY);

HoleT+=1;

AperatureTest();

TextOut(hdc, 30, 30, buffer, wsprintf(buffer, "loc(L,T) = %d, %d  
w=%d h=%d", HoleL, HoleT, HoleWidth, HoleHeight));

TextOut(hdc, 30, 70, buffer, wsprintf(buffer, "grey = %d",  
HoleGrey));

break;

097404-04304  
105707104



case VK\_LEFT:

BitBlt(hdc, 0, 0, cxClient, cyClient, viddc, 0, 0, SRCCOPY);

HoleL-=1;

AperatureTest();

TextOut(hdc, 30, 30, buffer, wsprintf(buffer, "loc(L,T) = %d, %d  
w=%d h=%d", HoleL, HoleT, HoleWidth, HoleHeight));

TextOut(hdc, 30, 70, buffer, wsprintf(buffer, "grey = %d",  
HoleGrey));

break;

case VK\_RIGHT:

BitBlt(hdc, 0, 0, cxClient, cyClient, viddc, 0, 0, SRCCOPY);

HoleL+=1;

AperatureTest();

TextOut(hdc, 30, 30, buffer, wsprintf(buffer, "loc(L,T) = %d, %d  
w=%d h=%d", HoleL, HoleT, HoleWidth, HoleHeight));

TextOut(hdc, 30, 70, buffer, wsprintf(buffer, "grey = %d",  
HoleGrey));

break;

TOP-TO-450

case VK\_F1:

BitBlt(hdc, 0, 0, cxClient, cyClient, viddc, 0, 0, SRCCOPY);

HoleHeight+=1;

AperatureTest();

TextOut(hdc, 30, 30, buffer, wsprintf(buffer,"loc(L,T) = %d, %d  
 w=%d h=%d", HoleL, HoleT, HoleWidth, HoleHeight));

TextOut(hdc, 30, 70, buffer, wsprintf(buffer,"grey = %d",  
 HoleGrey));

break;

case VK\_F2:

BitBlt(hdc, 0, 0, cxClient, cyClient, viddc, 0, 0, SRCCOPY);

HoleHeight-=1;

AperatureTest();

TextOut(hdc, 30, 30, buffer, wsprintf(buffer,"loc(L,T) = %d, %d  
 w=%d h=%d", HoleL, HoleT, HoleWidth, HoleHeight));

TextOut(hdc, 30, 70, buffer, wsprintf(buffer,"grey = %d",  
 HoleGrey));

break;

case VK\_F3:

BitBlt(hdc, 0, 0, cxClient, cyClient, viddc, 0, 0, SRCCOPY);

HoleWidth+=1;

AperatureTest();

TextOut(hdc, 30, 30, buffer, wsprintf(buffer,"loc(L,T) = %d, %d  
w=%d h=%d", HoleL, HoleT, HoleWidth, HoleHeight));

TextOut(hdc, 30, 70, buffer, wsprintf(buffer,"grey = %d",  
HoleGrey));

break;

case VK\_F4:

BitBlt(hdc, 0, 0, cxClient, cyClient, viddc, 0, 0, SRCCOPY);

HoleWidth-=1;

AperatureTest();

TextOut(hdc, 30, 30, buffer, wsprintf(buffer,"loc(L,T) = %d, %d  
w=%d h=%d", HoleL, HoleT, HoleWidth, HoleHeight));

09771011-012601

HoleGrey));  
TextOut(hdc, 30, 70, buffer, wsprintf(buffer, "grey = %d",

break;

case VK\_F5:

/\*

BitBlt(hdc, 0, 0, cxClient, cyClient, viddc, 0, 0, SRCCOPY);

HoleGrey-=10; //-darker

HoleGrey=(HoleGrey<0)? 0:HoleGrey; //if r>255 make it 255,  
else let it be

AperatureTest();

TextOut(hdc, 30, 30, buffer, wsprintf(buffer, "loc(L,T) = %d, %d  
w=%d h=%d", HoleL, HoleT, HoleWidth, HoleHeight));

TextOut(hdc, 30, 70, buffer, wsprintf(buffer, "grey = %d",  
HoleGrey));

\*/

BitBlt(hdc, 0, 0, cxClient, cyClient, viddc, 0, 0, SRCCOPY);

DarkThresh-=5;

DarkThresh));

TextOut(hdc, 30, 70, buffer, wsprintf(buffer, "thresh = %d",

break;

case VK\_F6:

/\*

BitBlt(hdc, 0, 0, cxClient, cyClient, viddc, 0, 0, SRCCOPY);

HoleGrey+=10;

HoleGrey=(HoleGrey>255)? 255:HoleGrey; //if r>255 make it  
255, else let it be

AperatureTest();

TextOut(hdc, 30, 30, buffer, wsprintf(buffer, "loc(L,T) = %d, %d  
w=%d h=%d", HoleL, HoleT, HoleWidth, HoleHeight));

TextOut(hdc, 30, 70, buffer, wsprintf(buffer, "grey = %d",  
HoleGrey));

\*/

BitBlt(hdc, 0, 0, cxClient, cyClient, viddc, 0, 0, SRCCOPY);

DarkThresh+=5;

TextOut(hdc, 30, 70, buffer, wsprintf(buffer, "thresh = %d",

FORGET THE TOP 200

DarkThresh));

break;

case VK\_SPACE:

//-----Get video format info and begin capture

//---1st get pointer to VideoFormat BITMAPINFO struct

pformatbmi

BITMAPINFO for buffer

dwsize=capGetVideoFormatSize(ghWndCap); //retrieve size of

global memory

hgout=GlobalAlloc(GHND,dwsize); //allocate that amount of

start

pformatbmiv=GlobalLock(hgout); //locks object, returns handle to

pformatbmi=(LPBITMAPINFO)pformatbmiv; //convert this  
pointer from void to bitmapinfo struct pointer

//LPBITMAPINFO pformatbmi;

//LPVOID pformatbmiv;

capGetVideoFormat(ghWndCap, pformatbmi, dwsize);

//valid BITMAPINFO struct is now at address pformatbmi

00000000-00000000-00000000-00000000

GlobalUnlock(hgout);

//-----CAPTURE!!

white to cover text etc

PatBlt(hdc, 0, 0, cxClient, cyClient, PATCOPY); //paint screen

//make capture window invisible

ShowWindow(ghWndCap,FALSE);

capPreviewScale(ghWndCap,0);//turn off scaling to capture  
window (may improve speed)

CursorValue=ShowCursor(0);//hide cursor

capCaptureSequenceNoFile(ghWndCap);//capture but don't save

//-----after capture ends (with escape key)-----

PatBlt(hdc, 0, 0, cxClient, cyClient, PATCOPY);

capPreviewScale( ghWndCap, 1);

ShowWindow(ghWndCap,TRUE);

CursorValue=ShowCursor(1);

09771011-012601  
T092T0-T0T260

break;

case VK\_RETURN: //only works after stopping capture w escape

CleanUp();

PostQuitMessage(0); //user clicked close, puts WM\_QUIT  
message in the message queue

break;

} //end switch

return 0;

case WM\_DESTROY: /\*terminate the program\*/

//WM\_CLOSE when processed by defwinproc generates a WM\_DESTROY msg.

//this is the last msg WinProc will get

CleanUp();

097401-01301  
105210-11012601





//spaces are omitted (ie not added to the DROP array)

//but the x loc of the next char is adjusted appropriately

//for readin - count line, figure out center and where line should start (left margin)

//for now - if line is too long, just omit rest.

int i;

int j=0;

int count=0;

int size = strlen(myLineText); //length of the string (character array) we're passing

for(i=0; i<size; i++) //loop through all the letters in this line

{

while(myLineText[j] == ' ') // this is to skip spaces

{

j++; //everytime there's a space j gets one more ahead of i

09771011-012501  
109210-101250

TOP TO BOTTOM

```
    }

    mLetterArray[i].ltr=myLineText[j];

    letter          mLetterArray[i].x=j*cxCaps;          //starting x loc of the

    mLetterArray[i].y=(rand() % 200)-200;    //starting y loc of the letter

    mLetterArray[i].rate=rand() % 3;          //random rate for letter 1-3?

    //letter[i].rate=2;

    LineInfos       mLetterArray[i].fadecolor=myLineColor;  //assign linecolor passed from

    count++;

    j++;

    }

    return count;

}
```

```
void SetToTop(COLORREF myColor,DROP mLetterArray[],int myLineLength)

{
```

```

int i;

for(i=0; i<myLineLength; i++)

{

    mLetterArray[i].y=(rand() % 200)-200;

    mLetterArray[i].fadecolor=myColor;

}

```

```

}

```

```

void Run(HWND hwnd)

```

```

{

```

```

    /*-----paint hdc(screen) with the update areas from memdc -----*/

```

```

    BitBlt(hdc, 0, 0, cxClient, cyClient, viddc, 0, 0, SRCCOPY);

```

```

}

```

```

bool CheckPixel(int X, int Y)

```

```

{

```

```

    bool yesorno=1;

```

T0920: T0T460

COLORREF colorref;

colorref=GetPixel(viddc,X,Y);

//0 is closer to dark, 255,255,255 = white

if(GetRValue(colorref)<DarkThresh && GetGValue(colorref)<DarkThresh &&  
GetBValue(colorref)<DarkThresh)

{

yesorno=0;

}

return yesorno;

}

void IncLocAndRgn(DROP myletter[], int mylinelength)

{

int i;

int deltadown;

int testdown;

00771011-012601  
T092T0-TT0T260

```
for(i=0; i<mylinelength; i++)
```

```
{
```

```
    if (myletter[i].y < (480-cyChar)) //letter is still on screen
```

```
    {
```

```
        //-----INCREMENT y values
```

```
        //there is the amount I want to move by, and the amount I want to
```

test

```
        //which is a little lower than the move amount
```

```
        deltadown=myletter[i].rate + (rand() % 3); //amount we want to
```

fall

```
        //testdown = myletter[i].y + deltadown; //where this would end up
```

would end up + offset

```
        testdown = myletter[i].y + deltadown+ cyChar-1; //where this
```

```
        //test color of pixel you're about to move to
```

```
        if(CheckPixel(myletter[i].x,testdown)) //1=white
```

```
            //myletter[i].y = testdown; //if ok, set y to this loc
```

```
            myletter[i].y += deltadown;
```

```
        else
```

```
        {
```

00771011-012201  
T092T0T0T0200

checkpix returns 0

```
while (CheckPixel(myletter[i].x,testdown)==0)//as long as
```

checkpixel with this value =1

```
testdown-=3;//subtract from testdown until
```

```
//deltadown-=3;
```

```
myletter[i].y = testdown-cyChar+1;
```

```
//myletter[i].y +=deltadown;
```

```
}
```

```
}
```

```
else
```

```
//-----OFF BOTTOM -- reset to top
```

```
myletter[i].y = 10;
```

```
//end for --gone through whole set of letters
```

```
}
```

097104:01604  
F09E10:F0F260

//-----Video Callback function

LRESULT CALLBACK VideoCallbackProc(HWND ghWndCap, LPVIDEOHDR lpVHdr)

{

int i;

int LineToStop;

int SecondLineToDrop;

if(!hwnd)

return FALSE;

//-----decompress, mirror and stretch incoming captured frame to vidc

DrawDibDraw(hdd,

backwardsdc,

0,0,

//MM\_TEXT client coordinates, of

the upper left corner of the destination rectangle

(\*pformatbmi).bmiHeader.biWidth,(\*pformatbmi).bmiHeader.biH

eight, //height and width

(LPBITMAPINFOHEADER)pformatbmi,

lpVHdr->lpData,

FORGET THE PAST





SRCCOPY);

//-----check if line is below a certain level

if(CheckLevel((ActiveLineIndex<gNumLines-1)? ActiveLineIndex+1:0)) //send this  
activelineindex+1,or wrap

{

//-----start active line FADING

LineInfos[ActiveLineIndex].fade=1;

//-----STOP active line -1 from FADING and FALLING (may have to reach back  
to top)

LineToStop=(ActiveLineIndex>0) ? ActiveLineIndex-1: gNumLines-1;

LineInfos[LineToStop].move=0; //stop moving

LineInfos[LineToStop].fade=0; //stop fading

SetToTop(LineInfos[LineToStop].color, Lines[LineToStop].dropline,  
LineLengths[LineToStop]); //reset x and y's for that line to top

//reset line

//-----INCREMENT active line number, and start new line FALLING

00771011012604  
F020204

```
SecondLineToDrop=(ActiveLineIndex<gNumLines-2)?  
ActiveLineIndex+2:((ActiveLineIndex<gNumLines-1)? 0:1);
```

```
//2
```

```
LineInfos[SecondLineToDrop].move=1; //turn on new line
```

```
ActiveLineIndex=(ActiveLineIndex<gNumLines-1)? ActiveLineIndex+1:0;
```

```
}
```

```
//move, fade and print lines that are both moving and fading
```

```
for (i=0; i<gNumLines; i++)
```

```
{
```

```
    if(LineInfos[i].move && LineInfos[i].fade)
```

```
    {
```

```
        IncLocAndRgn(Lines[i].dropline, LineLengths[i]);
```

```
        //move letters based on vidc image
```

```
        FadeText(Lines[i].dropline, LineLengths[i]);
```

```
        //fade text colors
```

```
        PrintText(Lines[i].dropline, LineLengths[i]);
```

097101-01301  
T09210-T07250

```
}  
}
```

ones) //move, and print lines that are moving (do this here so these lines are in front of fading

```
for (i=0; i<gNumLines; i++)
```

```
{
```

moving. if(LineInfos[i].move && (!LineInfos[i].fade))//assumes you won't fade w-out

```
{
```

```
IncLocAndRgn(Lines[i].dropline, LineLengths[i]);
```

```
//move letters based on viddc image
```

```
PrintText(Lines[i].dropline, LineLengths[i]);
```

```
}
```

```
}
```

```
AperatureFix(); //send shadow image to sheild camera
```

```
Run(hwnd); //bitblt to scrn
```

```
return (LRESULT) TRUE;
```

```
}
```

FOR THE TEST

void FadeText(DROP myLetter[], int myLineLength)//called once for each line that needs to be faded

{

int i;

COLORREF pixelcolor;

int scrnR, scrnG, scrnB;

int r, g, b;

UINT fadedir;

int fadeamount=5;

//I want things to desaturate as well

for (i=0; i<myLineLength; i++)

{

pixelcolor=GetPixel(viddc, myLetter[i].x + (cxChar/2), myLetter[i].y+(cyChar+1/2));

if(myLetter[i].y > 0 && (myLetter[i].fadecolor!=pixelcolor))

{

09771011-012601  
T092T0T0T0T0T0



0974041-043604  
T09T0-T0450

```
}
```

```
if(b!=scrnB)
```

```
{    //add or subtract fadeamount
```

```
    fadedir=(b>scrnB)? -1:1;
```

```
    b+=fadedir*fadeamount;
```

```
    b=(b>255)? 255:b; //if r>255 make it 255, else let it be
```

```
    b=(b<0)?0:b;
```

```
}
```

```
myLetter[i].fadecolor=RGB(r,g,b);
```

```
}
```

```
}//end for
```

```
}
```

```
void PrintText(DROP myLetter[], int myLineLength)
```

```
//called once for each line that needs to be printed
```

```
{
```

```
    int i;
```

```

    for (i=0; i<myLineLength; i++)
    {
        SetTextColor(viddc, myLetter[i].fadecolor);//could do a check here - only set if
necessary
        TextOut(viddc, myLetter[i].x, myLetter[i].y, buffer, wsprintf(buffer,"%c",
myLetter[i].ltr));
    }
}

```

```

bool CheckLevel(int myLine)
{
    int i=0;
    int numBelowLevel=0;

    while(i<LineLengths[myLine])
    {
        //loop through letter array for this line, increment belowLevel for all letters below
100 y coord
        if(Lines[myLine].dropline[i].y > 100)
            numBelowLevel++;

        i++;
    }

    if(numBelowLevel> (LineLengths[myLine]/2))

```

00771011-012501  
T032T0-TT0T2260



```
return 1;
```

```
else
```

```
return 0;
```

```
}
```

```
//-----
```

```
int FindDriver(char driverstring[80], int stringlength)
```

```
{
```

```
    char szDeviceName[80];           //string for camera names
```

```
    char szDeviceVersion[80];       //string for camera versions
```

```
    int wIndex;                      //camera driver index numbers
```

```
    int j;                           //for looping through driver string
```

```
    bool match=1;
```

```
    //-----test which video drivers are available
```

097104-01601  
FOUO

for (wIndex=0; wIndex < 10; wIndex++) //9 is the max index #

{

if(capGetDriverDescription  
(wIndex,szDeviceName,sizeof(szDeviceName),

szDeviceVersion,  
sizeof(szDeviceVersion)))

{

//loop through name to see if it's the QuickCam

for(j=0; j<(stringlength-1); j++)

if(szDeviceName[j]!=driverstring[j])

match=0;//if any letter doesn't match, flip this  
toggle

//break; //could use break?

if(match)

return wIndex; //didn't break - they must be equal

match=1;

}

}

//no matches

return 11;

09771041-042604

}

void CleanUp()

{

//-----disconnect driver

capDriverDisconnect(ghWndCap);

//-----clean up GDI stuff

DeleteDC(viddc);

DeleteDC(hdc);

DeleteObject(tempbitmap2);

DeleteDC(backwardsdc);

DeleteObject(BackwardsBitmap);

//-----close DrawDibLib

DrawDibClose(hdd);

067101101901

//-----be sure cursor value is set back to 0

if(CursorValue!=0)

{

while(CursorValue<0)

CursorValue=ShowCursor(1);

while(CursorValue>0)

CursorValue=ShowCursor(0);

}

}

void AperatureTest(){

HBRUSH oldbrush;

HPEN oldpen;

0974044-042504

```
oldpen=SelectObject(hdc,CreatePen(PS_SOLID,1,RGB(HoleGrey,HoleGrey,HoleGrey)));
//selects new brush into hdc
```

```
oldbrush=SelectObject(hdc,CreateSolidBrush(RGB(HoleGrey,HoleGrey,HoleGrey)));
//CONST LOGBRUSH *lplb
```

```
//RGB(HoleGrey,HoleGrey,HoleGrey)
```

```
Ellipse(      hdc, // handle to device context
           HoleL, // x-coord. of bounding rectangle's upper-left corner
           HoleT, // y-coord. of bounding rectangle's upper-left corner
           HoleL+HoleWidth, // x-coord. of bounding rectangle's lower-right
           HoleT+HoleHeight // y-coord. bounding rectangle's f lower-right
        );
```

```
DeleteObject(SelectObject(hdc,oldpen)); //selects oldbrush and deletes one we created.
```

```
DeleteObject(SelectObject(hdc,oldbrush));
```

```
//The ellipse is outlined by using the current pen and is filled by using the current brush.
```

00771011-012301

```
}
```

```
void AperatureFix(){
```

```
    HBRUSH oldbrush;
```

```
    HPEN oldpen;
```

```
    oldpen=SelectObject(viddc,CreatePen(PS_SOLID,1,RGB(HoleGrey,HoleGrey,HoleGrey)));  
    //selects new brush into hdc
```

```
        oldbrush=SelectObject(viddc,CreateSolidBrush(RGB(HoleGrey,HoleGrey,HoleGrey)));  
    //CONST LOGBRUSH *lp1b
```

```
        //RGB(HoleGrey,HoleGrey,HoleGrey)
```

```
    Ellipse(        viddc, // handle to device context
```

```
        HoleL, // x-coord. of bounding rectangle's upper-left corner
```

```
        HoleT, // y-coord. of bounding rectangle's upper-left corner
```

```
        HoleL+HoleWidth, // x-coord. of bounding rectangle's lower-right  
corner
```

09771041-01604  
T092T0-T0T260

HoleT+HoleHeight // y-coord. bounding rectangle's f lower-right  
corner

);

DeleteObject(SelectObject(viddc,oldpen)); //selects oldbrush and deletes one we created.

DeleteObject(SelectObject(viddc,oldbrush));

}

void DrawBlackRect(HDC whichdc, int left, int top, int right, int bottom){

HBRUSH oldbrush;

HPEN oldpen;

oldpen=SelectObject(viddc,CreatePen(PS\_SOLID,1,RGB(0,0,0))); //selects new brush  
into hdc

oldbrush=SelectObject(viddc,CreateSolidBrush(RGB(0,0,0))); //CONST LOGBRUSH  
\*lp1b

//RGB(HoleGrey,HoleGrey,HoleGrey)

09771041-012504  
T092T0="012504

```
Rectangle(    whichdc, // handle of device context

             left, // x-coord. of bounding rectangle's upper-left corner
             top, // y-coord. of bounding rectangle's upper-left corner
             right, // x-coord. of bounding rectangle's lower-right corner
             bottom // y-coord. of bounding rectangle's lower-right corner

            );
```

```
DeleteObject(SelectObject(viddc,oldpen)); //selects oldbrush and deletes one we created.
```

```
DeleteObject(SelectObject(viddc,oldbrush));
```

```
}
```

```
bool ReadInTextFromFile(LPCTSTR szFileName,LPVOID szBuffer,DWORD
maxcharstoread,LPDWORD lpcharsread){
```

```
HANDLE myHandle;    //returned by CreateFile
```

10927101-01601



bool bReadFileOK; //to test createfile (though also using formatMessage to do this)

bool myResult=0; //returned by ReadFile

myHandle=CreateFile(szFileName,

GENERIC\_READ,

FILE\_SHARE\_READ,

NULL,

OPEN\_EXISTING,

FILE\_ATTRIBUTE\_NORMAL,

NULL);

if(myHandle==INVALID\_HANDLE\_VALUE){

//----CreateFile returned an INVALID FILE HANDLE

//TextOut(hdc, 400, 10, buffer, wsprintf(buffer, "error in createfile"));

//--set our tracking var

bReadFileOK=FALSE;

09771011-01264

be closed

CloseHandle(myHandle); //readfile was successful so this needs to

}else{

//--ReadFile DID NOT PROCESS OK

bReadFileOK=FALSE;

}

}//end else for createfile

return bReadFileOK;

}

2025-01-01 10:00:00

[illegible]

```
//----CreateFile returned an VALID FILE HANDLE
```

```
myResult = ReadFile(myHandle, // handle of file to read
```

receives data LPVOID

bytes read

overlapped data

```
//--ReadFile PROCESSED OK
```

```
//TextOut(hdc, 10, 50, buffer, wsprintf(buffer,
"NumberOfBytesRead = %d", NumberOfBytesRead ));
```

```
strlen(szBuffer)); //TextOut(hdc, 10, 130, buffer, wsprintf(buffer, "textlength= %d",
```